

DOCUMENT RESUME

ED 460 001

SE 065 492

AUTHOR Abraham, Paul
TITLE Using the TI-92+ Calculator as a Tool for Illustrating Programming Concepts.
PUB DATE 1999-11-00
NOTE 6p.; Paper presented at the Annual International Conference on Technology in Collegiate Mathematics (12th, San Francisco, CA, November 4-7, 1999).
AVAILABLE FROM For full text:
<http://archives.math.utk.edu/ICTCM/EP-12/C15/pdf/paper.pdf>.
PUB TYPE Guides - Classroom - Teacher (052) -- Opinion Papers (120) -- Speeches/Meeting Papers (150)
EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS College Curriculum; *Computer Science Education; *Computer Uses in Education; Educational Strategies; *Graphing Calculators; Higher Education; *Programming Languages; Science Education
IDENTIFIERS *MAPLE Computer Algebra System

ABSTRACT

This paper describes how the TI-92+ calculator can be valuable in illustrating basic and advanced programming concepts in introductory computer science courses. Descriptions of the scope of the curricula for the courses in which the TI-92+ is used are provided. The TI-92+ programming language is particularly useful in demonstrating the progression in program development from pseudocode to code. Examples using MAPLE code are employed. Programming examples using TI-92+ are also included. (DDR)

Using the TI-92+ Calculator as a Tool for Illustrating Programming Concepts

By Paul Abraham
Kent State University Stark Campus

Introduction

As a mathematics professor who occasionally teaches introductory computer science courses, I have found the TI-92+ calculator to be valuable in illustrating basic and advanced programming concepts, especially when mathematics is an important underlying issue. The computer science courses I teach are an Introduction to Computer Science (CS 10051) and an Introduction to Object-Oriented Programming (CS 23021) at Kent State University Stark Campus.

CS 10051 surveys major areas of computer science like development of pseudocode and elementary structured (C++) programs, analysis of algorithms, data representation, basic computer architecture and assembly language and programming paradigms. The course lays the foundation for entry into many other computer science courses and is also the required computer science class for mathematics majors. Since the TI-92+ programming language is so natural it was particularly useful in demonstrating the progression in program development from pseudocode to code. A small segment of the course allows for a discussion of symbolic programming. As with the adopted text [1] for the course I presented a few examples of *Maple* code, but I also continued with TI-92+ examples to give students an idea of the current power of hand-held technology.

CS 23021 follows CS 10051 and is the first programming-directed course in the computer science major. As the title suggests, course discussions and student assignments progress to involve critical and varied use of objects. In particular string objects, vector objects (which behave much like TI-92+ lists), rational number objects and geometrical window objects like *RectangleShape* objects play a major role in the course and in the adopted text [2]. In this course I gave a few TI-92+ demonstrations which incorporated lists into symbolic programming to show how lists could be used to solve substantial mathematical problems. The function *rsolve()* (discussed in detail below) which determines the regions on the number line that pass/fail a given rational inequality was such an example. With College Algebra (Math 11011) as a prerequisite for CS 23021, students could certainly appreciate the significance of *rsolve()*. Since TI-92+ fractions and imaginary numbers are represented in standard form, I also used the TI-92+ to motivate the development of C++ rational and complex number objects.

Example Functions: *rsolve()*, *mysort()*, *ispos()* and *gsolve()*

PERMISSION TO REPRODUCE AND
DISSEMINATE THIS MATERIAL HAS
BEEN GRANTED BY

P. Abraham

TO THE EDUCATIONAL RESOURCES
INFORMATION CENTER (ERIC)

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION
CENTER (ERIC)

This document has been reproduced as
received from the person or organization
originating it.

☐ Minor changes have been made to
improve reproduction quality.

• Points of view or opinions stated in this
document do not necessarily represent
official OERI position or policy.

BEST COPY AVAILABLE

In this section the functions `rsolve()`, `mysort()` and `ispos()` are presented in entirety and a fourth function `gsolve()` is discussed. The code for all of these functions is available from the author and perhaps the Calculator Program Archive [4].

The function `rsolve()` solves any given rational inequality. The solution returned by `rsolve()` is only partial in the sense that it determines the regions on the number line that pass/fail the inequality, but does not decide which endpoints should be included. The function `rsolve()` returns the solution as a list which mimics the bottom line of sign chart solutions commonly constructed in textbooks, e.g. see [3].

The function `gsolve()` solves any given rational inequality completely. It returns the final answer using standard interval notation, singleton sets or the empty set as needed. The code for `gsolve()` is somewhat more tedious than `rsolve()` and will not be presented here since it occupies several pages. Runs of both `rsolve()` and `gsolve()` follow:

F1	F2	F3	F4	F5	F6
Algebra	Calc	Other	PrgmIO	Clean Up	
<pre> *rsolve(x<4,x) (-e true -2 false 2 true =>) *rsolve(x/(x-3)>0,x) (-e true 0 false 3 true =>) *rsolve(x^2/(x-3)>0,x) (-e false 0 false 3 true =>) rsolve(x^2/(x-3)>0,x) </pre>					
PRGCFP	AND AUTO	FUNC	3/20		

F1	F2	F3	F4	F5	F6
Algebra	Calc	Other	PrgmIO	Clean Up	
<pre> *gsolve(x^2/(x-3)>0,x) " (0)U(3,e)" *gsolve(1/x^2<0,x) () *gsolve(x-4/(x^2-2x-3)<0,x) "(-e,-1)U(3,4)" gsolve((x-4)/(x^2-2x-3)<0,x) </pre>					
PRG	AND AUTO	FUNC	3/20		

The comments in the `rsolve()` function code below explain fully how the function works. As the output of `rsolve()` illustrates, the solution is a list which contains different data types: extended real numbers and Boolean values. The ability for a list to contain values of different base types is extremely powerful and rare in common languages.

```
rsolve(ineq,var)
Func
```

"This function determines the regions on the number line that satisfy the given rational inequality ineq in variable var.

"The result answ is returned as an ordered list of numbers with true or false between adjacent numbers. A true between adjacent numbers indicates the corresponding region satisfies the inequality, else a false appears.

Local oneside,combined,top,bottom

"The left and right sides of ineq will be combined with a common denominator into one side called combined. 0 is understood to be the new right-hand side. top is the numerator of combined and bottom is the denominator of combined.

```
left(ineq)-right(ineq)=oneside
comDenom(oneside,var)=combined
getNum(combined)=top
getDenom(combined)=bottom
```

Local s,z,endpt,answ

"These variables represent lists. s will hold the zeros of bottom, z the zeros of top, and endpt will be the ascending sorted list sUz of endpoints.

```

zeros(top,var)»z
zeros(bottom,var)»s
augment(s,{0,1})»s
z»endpt
augment(endpt,s)»endpt

```

```
mysort(endpt)»endpt
```

"The code for mysort() occurs later in this article. It was necessary to write mysort since a function can not call the built-in ascending sort program.

Local a,b,i,testv

"These variables are used to construct answ. a represents the left endpoint for the current region, b represents the right endpoint, i maintains the current position in endpt and testv represents the test value for the current region.

"answ begins with left endpoint a -. Looping ends when a reaches .

```

1»i
a»a
{0,1}»answ

```

```

While a
endpt[i+1]»b

```

" this block of code picks a test value appropriately.

```

If a=0, Then
  If b=, Then
    0»testv
  Else
    b-1»testv
  EndIf
Else
  If b=, Then
    a+1»testv
  Else
    (a+b)/2»testv
  EndIf
EndIf

```

" The next two lines insert true or false and the right endpoint into answ. testv is plugged into ineq to decide between true or false.

```

augment(answ,{ineq|var=testv})»answ
augment(answ,{b})»answ

```

" Finally, the right endpoint becomes the new left endpoint and the current position in endpt increases.

```

b»a
i+1»i
EndWhile

```

```

Return answ
EndFunc

```

As you may have noticed rsolve() requires a list to be sorted and calls the function mysort(). The function mysort() is a routine bubble sort. The code for mysort() follows:

```

mysort(l)
Func
  This function bubble sorts list l.

  Local i,j,n,temp
  dim(l)»n
  1»i
  While i<n
    1»j
    While j«n-i
      If l[j]>l[j+1] Then
        l[j]»temp
        l[j+1]»l[j]
        temp»l[j+1]
      EndIf
      j+1»j
    EndWhile
    i+1»i
  EndWhile
  Return l
EndFunc

```

We conclude with the code for the function ispos() that illustrates another nice feature of lists on the TI-92+: the ability to apply a function or operator across a list. In calculus this feature could be used for example to differentiate a list of functions in one step, e.g., $d(\{x, x^2, x^3\}, x)$ produces the list $\{1, 2x, 3x^2\}$. Similarly, the function ispos() determines which list entries are positive without explicit looping through the list and returns a list of true/false outcomes. The call ispos($\{1, -2, 0\}$) produces $\{\text{true}, \text{false}, \text{false}\}$.

```

ispos(l)
Func
  Local z,n
  dim(l)»n
  n is the size of l
  newList(n)»z
  Created a list z of size n and by default filled it with 0's. The comparison of l and z entries is done next
  coordinate-wise and the list of results is returned
  Return l>z
EndFunc

```

References:

- [1] J.P. Cohoon and J.W. Davidson, **C++ Program Design: An Introduction to Programming and Object-Oriented Design**, 2nd edition, McGraw-Hill Inc., 1999.
- [2] G.M. Schneider and J.L. Gersting, **An Invitation to Computer Science**, 2nd edition, Brooks/Cole Publishing, 1999.
- [3] M. Sullivan, **College Algebra**, 5th edition, Prentice Hall, 1999.
- [4] Texas Instruments Inc., Calculator Program Archive, Texas Instruments Inc., <<http://www.ti.com/calc/docs/arch.htm>>, 22 December 1999.

Author's Postal Address:

KSU Stark Campus
6000 Frank Avenue NW
Canton OH 44720

Email Address: pabraham@stark.kent.edu